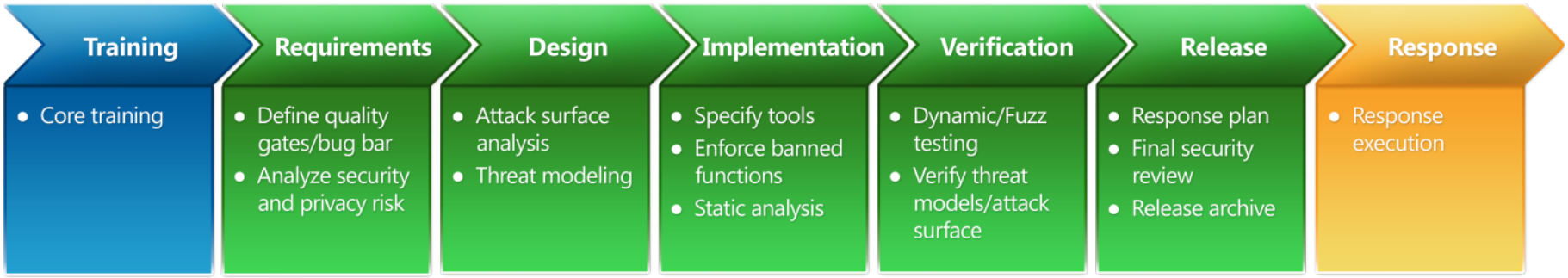
DEVELOPMENT PROCESS FOR SECURE SOFTWARE

Development Process

- Emphasis on „building secure software” as opposed to „building security software”
- Major methodologies
  - Microsoft's Security Development Lifecycle
  - Cigital's Security touchpoints
  - OWASP CLASP
- Building Security In Maturity Model – BSIMM

Microsoft SDL



OWASP

- Open Web Application Security Project
- Collects resources for web development
  - Top ten security flaws
  - Various security testing tools
  - Various guides
    - e.g. code review guide
  - CLASP – Comprehensive, Lightweight Application Security Process

CLASP

- Goal: move security concerns into the early stages of the software development lifecycle, whenever possible
- Set of process pieces that can be integrated into any software development process
  - Roles
  - Activities that can be performed
  - Vulnerability classifications
  - Vulnerability use cases

CLASP Best Practices

- Institute awareness programs
- Perform application assessments
- Capture security requirements
- Implement secure development practices
- Build vulnerability remediation procedures
- Define and monitor metrics
- Publish operational security guidelines

Institute Awareness Programs

- People should consider security to be an important project goal
- Train all team members
- Make people aware of security setting
- Institute accountability for security issues
- Appoint a project security officer
- Institute rewards for handling of security issues

Perform Application Assessments

- Security analysis of requirements and design
  - Threat modelling
- Source-level security review
- Security tests

Capture Security Requirements

- Treat security requirements same way as functional requirements
- Define security policy
- Identify attack surface
- Identify resources and trust boundaries
- Identify misuse cases
- Specify operational environment

Implement Secure Development Practices

- Annotate classes with security properties
- Apply principles of secure design
- Manage resources
- Manage contracts and interfaces

Build Vulnerability Remediation Procedures

- Address reported security issues
- Manage security issue disclosure process

Define and Monitor Metrics

- Select metrics
- Collect data
- Evaluate results

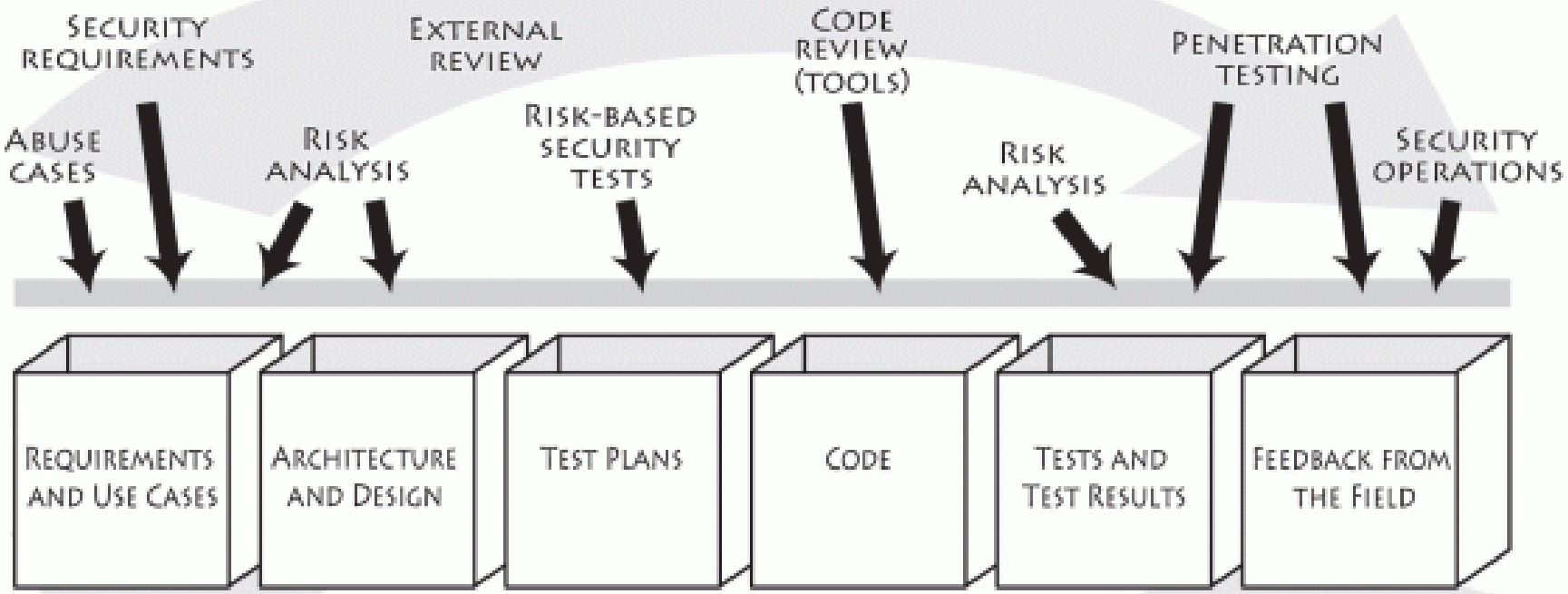
Publish Operational Security Guidelines

- Build operational security guide
- Specify database security configuration

Security Touchpoints

- Gary McGraw, Cigital
- More oriented towards programming
  - „All software projects produce at least one artifact: source code”
- „Software Security: Building Security In”, by Gary McGraw

Security Touchpoints (2)



Code Review with a Tool

- Aim: catching implementation bugs early
- Tool helps to achieve good code coverage
- Aim for good, not perfect

Architectural Risk Analysis

- First step: create description of architecture
  - Start with one page
  - Forest-level view
- Attack resistance
  - Use checklists of known attacks
  - Example: Microsoft STRIDE
    - **S**poofing, **T**ampering, **R**epudiation, **I**nfo disclosure, **D**enial of service, **E**levation of privilege

Architectural Risk Analysis (2)

- Ambiguity analysis
  - Discover new risks
  - Find unclear parts in how the system works
  - Trust, data sensitivity, threat models
- Weakness analysis
  - Impact of external software dependencies
  - Platform (hardware, OS)
  - Frameworks
  - Called services

Architectural Risk Analysis (3)

- Combine risks and consider business impact
- Rank risks
- Find solutions

Penetration Testing

- Use the source (white-box testing)
  - Otherwise people send time on reverse-engineering system
- Apply business priorities
  - Logic flaw vs. XSS flaw
  - XSS is important if it contributes towards compromising business logic

Penetration Testing (2)

- Use in-house QA department
  - They already know the system
  - Use tools and training to add security testing skills
- Test more than once
- Incorporate the findings back into development

Risk-Based Security Tests

- Test based on priorities
  - Architectural risks
  - Risks discovered during code review
- Test malicious input
  - Use fuzzing tool

Abuse Cases

- Security is not a set of features
- How system should react to illegitimate use
- Like use cases, but with malicious users

Common Attack Taxonomies

- 7 Pernicious Kingdoms; McGraw
- 19 Deadly Sins; Howard, LeBlanc, Viega
- 48 Attack Patterns; McGraw/Hoglund
- Common Weakness Enumeration
  - http://cve.mitre.org/cwe

Seven Pernicious Kingdoms

- Input Validation and Representation
- API Abuse
- Security Features
- Time and State
- Error Handling
- Code Quality
- Encapsulation
- *Environment*

BSIMM

- Building Security In Maturity Model
- Empirical approach to secure software design
- Gathered data from nine large-scale software security initiatives
  - Tech companies (Adobe, Google, Microsoft, QUALCOMM)
  - Financial companies (DTCC, Wells Fargo)
- Added only activities seen in the real world
- Actual practices, not best practices

Software Security Framework

- 12 practices in 4 domains
- Each practice consists of activities
- 110 activities in total
- Each activity at least in one company
- No company that does it all

SSF

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Practices

- Each practice contains activities
- Divided to three levels
- Example practice: code review

Code Review Level 1

- Description: SSG does code review
- Create a top N bugs list
- Have SSG perform *ad hoc* review
- Establish coding labs or office hours focused on review

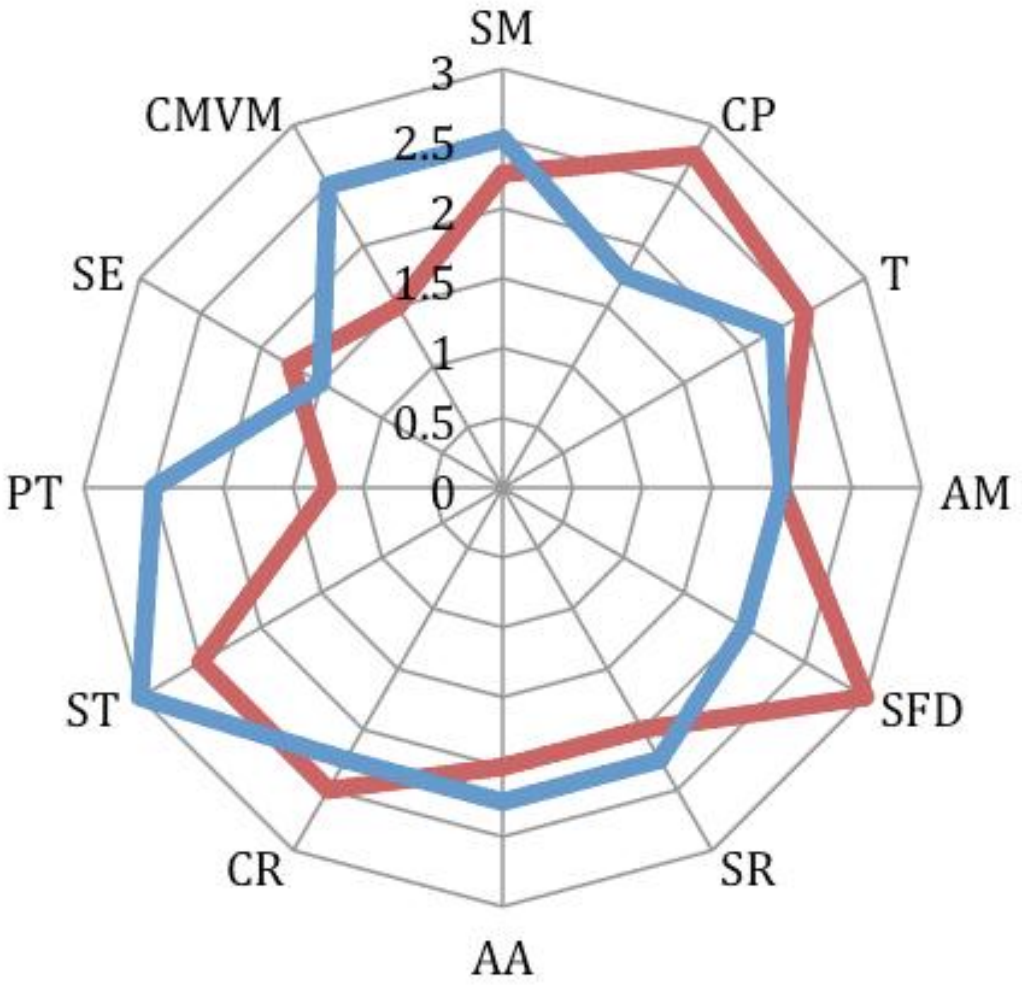
Code Review Level 2

- Description: enforce standards though mandatory automated code review and centralized reporting
- Use automated tools along with manual review
- Enforce coding standards
- Make code review mandatory for all projects
- Use centralized reporting
- Assign tool mentors

Code Review Level 3

- Description: Build an automated code review factory with tailored tools
- Use automated tools with tailored rules
- Build a factory
- Build capability for eradicating specific bugs from entire code base

Spider Chart



In Conclusion...

Final Words

- Don't do anything just because somebody else does
- Apply the scientific method
  - „*a method of procedure that has characterized natural science since the 17th century, consisting in systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses.*”